Paper CC22

Oh No, a Zero Row: 5 Ways to Summarize Absolutely Nothing

Stacey D. Phillips, i3 Statprobe, San Diego, CA
Gary Klein, i3 Statprobe, San Diego, CA

ABSTRACT

SAS® is wonderful at summarizing our data, including creating frequency counts and percentages. However, sometimes, what *isn't* in the data is just as important as what *is* in the data. Unfortunately, it is not so easy to get SAS to summarize what isn't there, e.g., how can a PROC FREQ count data points that do not exist in the data? This paper will explore five different methods of creating summaries that will contain a count of zero.

INTRODUCTION

Frequently the role of the SAS programmer is to present summaries of information from various sources. For example, in the pharmaceutical industry, the programmer may have to summarize all of the demographics that appear on a case report form. However, when the data contains a small population or there is something obscure on the CRF which no subject in the data fulfills, the summarization of all the points on the CRF becomes difficult. In another example, a statistician may want to see all of the values on the CRF in a table even if no subject in the data reported that characteristic. In these cases we are interested in the fact that no one is actually in the data---or as we call it here, a zero row. The goal of this paper is to present five different examples of how to get SAS to summarize those zero rows for us, that is, summarize records that aren't there.

INITIAL DATA

For our examples, we will create a dataset with some categories missing that we will need to summarize later. We will count the number of subjects with the different combinations of brown, blonde, and red hair with brown, blue and hazel eyes. So we will begin by creating the following dataset and formats:

```
proc format;
   value $ hair "Br" = "Brown"
                "Bl" = "Blonde"
                "Re" = "Red"
                ;
   value $ eyes "Br" = "Brown"
                "Bl" = "Blue"
                "Ha" = "Hazel"
                ;
quit;

data temp;
   attrib col1 label = 'Hair'
               format = $hair.

          col2 label = 'Eyes'
               format = $eyes.
               ;

   col1 = "Br"; col2 = "Br"; output;
   col1 = "Br"; col2 = "Bl"; output;
   col1 = "Bl"; col2 = "Br"; output;

run;
```

Below is a simple PROC PRINT of our data:

Hair	Eyes
Brown	Brown
Brown	Blue
Blonde	Brown

It should be apparent that the combination of blonde hair with blue eyes is missing, as well as all different combinations with

red hair and hazel eyes. We will need to summarize every combination.

TECHNIQUES TO SUMMARIZE MISSING DATA

METHOD 1 – PROC FREQ USING A DUMMY HARD-CODED DATASET

```
title3 "Method 1: PROC FREQ using hard-coded dummy dataset";
title4 "Benefits: No formats necessary.";
title5 "Limitations: Need to create a dummy dataset - susceptible to errors if any changes";

proc freq data=temp noprint;
   table col1 * col2 /out=way1a(drop=percent);
run;

data dummy;
    do col1 = "Br", "Bl", "Re";
       do col2 = "Br", "Bl", "Ha";
          output;
       end;
    end;
run;

proc sort data=way1a;
   by col1 col2;
run;

proc sort data=dummy;
   by col1 col2;
run;

data way1b;
   merge way1a dummy;
   by col1 col2;

   if count = . then count = 0;

run;

proc print data=way1b l noobs;
run;
title3;
```

Hair	Eyes	Frequency Count
Blonde	Blue	0
Blonde	Brown	1
Blonde	Hazel	0
Brown	Blue	1
Brown	Brown	1
Brown	Hazel	0
Red	Blue	0
Red	Brown	0
Red	Hazel	0

Discussion:
In this example, we use simple output statements to create a blank record for each possible combination of eye and hair color. After using PROC FREQ to create a dataset with the counts of actual data, we merge this dataset with the dummy dataset and have a complete set of frequencies for every possible combination of hair and eye color. The biggest advantage to this method is that it is simple and requires no formats. One large disadvantage, however, is that the programmer needs to be aware of all of the possible combinations before programming. It could become a maintenance nightmare if the possible values change.

METHOD 2 – PROC FREQ USING THE SPARSE OPTION

```
title3 "Method 2: PROC FREQ using sparse option" ;
title4 "Benefits: Simple Code.  No formats needed";
title5 "Limitations: - Need at least one from each row" ;

proc freq data=temp noprint;
   table col1 * col2 /out=way2(drop=percent) sparse;
run;

proc print data=way2 l noobs;
run;
title3;
```

Hair	Eyes	Frequency Count
Blonde	Blue	0
Blonde	Brown	1
Brown	Blue	1
Brown	Brown	1

Discussion:
Again in this example we use PROC FREQ to create a dataset that includes the frequencies of the various combinations of hair and eye color. This time, however, we use the "sparse" option in SAS. If we use the TEMP dataset from the first example there are 2 options for hair color: blond and brown and 2 options for eye color: blue and brown. Using the sparse option in PROC FREQ, SAS outputs a record for every possible combination that could potentially occur in the data rather than just the combinations that do occur. For example, there is no record of blonde hair and blue eyes in the data, but SAS lists it as a possible combination because blond hair and blue eyes occur in other data points. The sparse option is convenient to use and allows for simpler code. A glaring limitation is that the sparse option will only summarize what it sees in the data. So although we know hazel eyes and red hair are options from the CRF, SAS does not know this and these two characteristics are left off of the frequency counts.

METHOD 3 – PROC FREQ USING AN AUTOMATED DUMMY DATASET

```
title3 "Method 3: PROC FREQ with an automated dummy dataset";
title4 "Benefits: Automates all values from formats - less chance for error";
title5 "Limitations: Complicated Code, also need to know which format to apply";
title6 "(Could find formats using proc contents or sashelp - but it could get overly
complicated)";

proc freq data=temp noprint;
   table col1 * col2 /out=way3a(drop=percent);
run;

proc format cntlout=formats;
run;

proc sql;
   create table dummy as
      select a.start label="Hair" format=$hair. as col1,
             b.start label="Eyes" format=$eyes. as col2
      from formats(where=(fmtname='HAIR')) as a,
           formats(where=(fmtname='EYES')) as b;

   create table way3b as
      select a.col1,
             a.col2,
             coalesce(b.count, 0) as count
      from dummy as a left join way3a as b
   on a.col1 = b.col1 and a.col2 = b.col2;

quit;


proc print data=way3b l noobs;
run;
```

```
title3;
```

Hair	Eyes	COUNT
Blonde	Blue	0
Blonde	Brown	1
Blonde	Hazel	0
Brown	Blue	1
Brown	Brown	1
Brown	Hazel	0
Red	Blue	0
Red	Brown	0
Red	Hazel	0

Discussion:
This example automatically uses PROC SQL to create a dummy dataset based on the values of formats specified by the programmer. Then, using a combination of PROC SQL and the "coalesce" function, SAS joins the dataset with the counts (created from the PROC FREQ) with the dummy dataset and fills in a count of zero where an actual count from the data does not exist. This method is great because it is automatic and based on the formats, but is problematic because the code is rather complicated.

METHOD 4 – PROC MEANS USING "COMPLETETYPES" OPTION

```
title3 "Method 4: PROC MEANS using completetypes ";
title4 "Benefits: Simple code";
title5 "Limitations: Need at least one from each row";

proc means data=temp completetypes noprint nway;
   class col1 col2;
   output out=way4(rename=(_freq_=count) drop=_type_);
run;

proc print data=way4 l noobs;
run;
title3;
```

Hair	Eyes	COUNT
Blonde	Blue	0
Blonde	Brown	1
Brown	Blue	1
Brown	Brown	1

Discussion:
In this example, we use a method that is very similar to using the sparse option in PROC FREQ but instead this time we use PROC MEANS. Using PROC MEANS on dataset TEMP and the "completetypes" option, we get an output dataset that includes all possible combinations that could potentially occur in the data in addition to combinations that do occur.. So, even though there is no record with blonde hair and blue eyes, the completetypes option includes this combination because blonde hair and blue eyes occur elsewhere in dataset TEMP. Like the sparse option in PROC FREQ, the completypes option is very simple to use. Similar again to sparse, there must be at least one occurrence of a value for completypes to summarize appropriately.

METHOD 5 – PROC MEANS USING "COMPLETETYPES" AND THE "PRELOADFMT" OPTION

```
title3 "Method 5: PROC MEANS using completetypes and preloadfmt";
title4 "Benefits: Automated to get all records from formats and do not necessarily need to
know format if assigned";
title5 "Limitations: Needs formats";

proc means data=temp completetypes noprint nway;
   class col1 col2 /PRELOADFMT ;
   output out=way5(rename=(_freq_=count) drop=_type_);
run;

proc print data=way5 l noobs;
run;
```

```
title3;
```

Hair	Eyes	COUNT
Blonde	Blue	0
Blonde	Brown	1
Blonde	Hazel	0
Brown	Blue	1
Brown	Brown	1
Brown	Hazel	0
Red	Blue	0
Red	Brown	0
Red	Hazel	0

Discussion:
In the original dataset TEMP, the formats for both hair and eye color were assigned to each variable using ATTRIB statements. The PRELOADFMT option in PROC MEANS uses these assigned formats to determine what the possible combinations of values could be. Advantages to this method include simplicity of use and the fact there is no requirement to have at least one occurrence of a value in the data. A disadvantage is that this method only works if you are using formats in combination with your data. You don't necessarily need to know what the format values are, but you do need to be sure that the formats are assigned to the variables you are trying to summarize.

CONCLUSION

When producing summary tables in the pharmaceutical industry, it is frequently important to summarize what is not there as well as what is there. In this paper we've discussed five separate ways to accomplish this task and which method you choose depends on the complexity and characteristics of your data. Whichever method you choose, you should now be armed with the knowledge and the ability to summarize nothing!

CONTACT INFORMATION

Your comments and questions are valued and encouraged.
Contact the authors at:

Gary Klein
i3 Statprobe
10052 Mesa Ridge Court
Suite 200
San Diego, CA 92121
Work Phone: (858) 597-1000 ext. 2460
Email: gary.klein@i3statprobe.com

Stacey D. Phillips
i3 Statprobe
10052 Mesa Ridge Court
Suite 200
San Diego, CA 92121
Email: stacey.phillips@i3statprobe.com